Jarno Laine

Laravel-ohjelmistokehys nykyaikaisessa ohjelmistokehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

30.4.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Jarno Laine Laravel-ohjelmistokehys nykyaikaisessa ohjelmistokehityksessä 24 sivua 30.4.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja	Lehtori Juha Kämäri

Tämän opinnäytetyön tarkoituksena oli tutustua Laravel 5.2 -ohjelmistokehyksen ominaisuuksiin ja sen tuomiin mahdollisuuksiin nykyaikaisessa ohjelmistokehityksessä. Projektia varten luotiin esimerkkiohjelmistona urheiluseurojen tarpeita vastaava tilavarausohjelmisto, jonka avulla mallinnettiin ohjelmiston kehityksen eri vaiheita.

Laravel valittiin tähän tarkoitukseen sen saavuttaman suuren suosion takia PHP-yhteisössä. Laravel toteuttaa hyvin MVC-mallin mukaista web-ohjelmointia ja ohjelmistokehyksen ominaisuuksien tunteminen suunnittelumalleiltaan auttaa ymmärtämään myös muita ohjelmistokehyksiä ja niiden toimintaa.

Työn lopputuloksena toteutettiin ohjelmisto, joka vastaa urheiluseuroille merkittävien tapahtumien ylläpidon tarpeita ja josta pystytään raportoimaan laskutusta tai tuntien laskemiseen perustuvaa dataa.

Avainsanat	MVC, Laravel, Migraatio, Tietokanta, PHP

Author(s) Title Number of Pages Date	Jarno Laine Laravel framework in modern web development 24 pages 30 April 2017
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Specialisation option	Software engineering
Instructor(s)	Juha Kämäri, Senior Lecturer

Main goal of this thesis was to have introduction to Laravel 5.2 framework and in its usability in modern web development by developing a software to sports clubs.

Laravel was chosen for its big popularity in PHP community. Laravel uses MVC-pattern and the same principals are used in other popular frameworks like Symfony and Yii. Understanding these design choices helps to develop software with other frameworks.

Software responds needs of sports clubs resource planning. Sports clubs can mark scheduled events to calendar and have reports for the events. Software can calculate costs and hours used in training for specific calendar that corresponds to physical location.

Keywords	MVC, Laravel, Migraatio, Tietokanta, PHP

Sisällys

Lyhenteet

ORM	Object-relational mapping. Oliomallin mukaisen esityksen kuvaus relaatiomallin mukaiseksi esitykseksi.
TKHJ	Tietokannan hallintajärjestelmä. Ohjelmisto, jonka avulla hallinnoidaan tietokantoja.
MVC	Model View Controller. Yleinen suunnittelumalli, jota käytetään nykyaikaisessa ohjelmistokehityksessä.
REPL	Read-eval-print loop. Työkalu, jolla voidaan olla vuorovaikutuksessa ohjelmiston kanssa reaaliajassa.
WAMP	**W**indows **A**pache **M**ySQL **P**hp, ohjelmistokokonaisuus, joka toimii palvelinympäristönä windows-käyttöjärjestelmälle.

1 Johdanto

Laravel on PHP-skriptikieleen pohjautuva ohjelmistokehys (engl. Software framework), joka tarjoaa web-sovelluksiin MVC-arkkitehtuurin perustuvan ohjelmistokehityspohjan. Laravel on Taylor Otwellin kehittämä ohjelmistokehys, jonka ideana on tarjota yksinkertainen ja helposti omaksuttava rakenne ohjelmistolle, jossa on paljon ominaisuuksia niin aloitteleville kuin kokeneille ohjelmistokehittäjillekin. (1.)

Nykyaikaisessa web-ohjelmistoissa käytetään hyvin paljon erilaisia ohjelmistokehyksiä, jotka tarjoavat nopeaan kehitykseen erinomaiset mahdollisuudet. Tarkoituksena ohjelmistokehyksissä on tarjota perustoiminnallisuudet valmiissa paketissa. Laravelia ei ole kehitetty täysin tyhjästä, vaan siinä käytetään paljon mm. Symfony-ohjelmistokehyksen komponentteja.

Insinöörityön tarkoituksena on tutustua Laravelin rakenteeseen käytännön esimerkin pohjalta. Insinöörityössä on toteutettu tilanvaraukseen pohjautuva ohjelmisto urheiluseuroille, jonka tarkoituksena on tutustua ohjelmistokehyksen tarjoamiin mahdollisuuksiin.

Työ sisältää myös teoriaosuuden, jossa tarkastellaan lopullisen ohjelmiston kannalta tärkeitä konsepteja ohjelmistokehyksestä.

2 Laravel ohjelmistokehyksen käyttöönotto

Tässä osiossa kuvataan Laravelin asentamiseen ja konfiguroimiseen vaadittavat toimenpiteet.

2.1 Asennus

Laravel asettaa ohjelmistokehyksen toiminnalle ja asennettavalle palvelimelle seuraavat rajoitteet:

- PHP >= 5.5.9

- OpenSSL PHP -lisäosa
- PDO PHP -lisäosa
- Mbstring PHP -lisäosa
- Tokenizer PHP -lisäosa.

Laravel suositellaan asennettavaksi heidän omalle räätälöidylle virtuaaliympäristölleen, joka kantaa nimeä Homestead. Homestead perustuu Vagrant-ohjelmistoon, joka voidaan asentaa esimerkiksi VM Ware tai VirtualBox virtuaaliympäristöön. Vagrant-ympäristön tarkoituksena on tarjota helposti siirrettäviä virutaaliympäristöjä, jonka tarkoituksena on mm. varmistaa tuotantoympäristön ja kehitysympäristön samankaltaisuus ja toiminnallisuus.

Tässä työssä asennettiin Laravel perinteisesti testiympäristöön käsin. Käyttöjärjestelmänä toimi Linuxin Ubuntu Server 16.04 LTS:llä. Asennus tehtiin projektin aluksi Windows ympäristöön WAMP-ohjelmistokokonaisuuden päälle, josta se myöhemmin siirettiin Ubuntu palvelimelle. WAMP-ohjelmistokokonaisuudessa palvelinohjelmana toimii Apache, mutta Linuxia käyttävälle tuotantoserverille asennettiin Nginx. PHP-versiona projektissa toimi joulukuussa 2015 julkaistu PHP 7.0. (3) Toimiakseen tilanvarausohjelmisto vaatii edellä mainitun listan lisäksi tietokannan, johon tietoa tallennetaan. Tässä projektissa käytettiin MySQL-tietokantaa. Laravelin muita MySQL lisäksi tukemia tietokantoja ovat (4.)

- Postgres
- SQLite
- SQL Server.

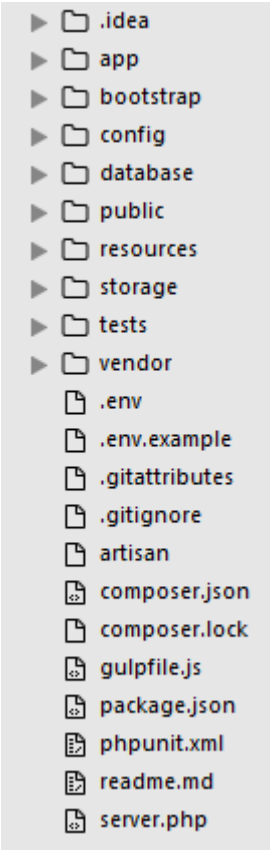
Jätän tässä insinöörityössä nämä komentorivillä suoritetun palvelimen peruskonfiguraation kuvaamatta. Kun perusympäristö on saatu konfiguroitua, voidaan aloittaa Laravelin asennus palvelinohjelmiston tiedostojuureen. Laravel asennetaan käyttäen Composer-pakettienhallintaohjelmaa. Laravelin asennus onnistuu antamalla Composerille komento: (2)

```
composer global require "laravel/installer"
```

Tämän jälkeen Laravel-projektin luonti onnistuu antamalla komento, jossa {Aplikaation nimi} vaihdetaan oikealla projektinimellä:

```
laravel new {Aplikaation nimi}
```

Kun projekti on luotu, sekä versionhallinta on asennettu, puhdas asennus näyttää seuraavalta:



Kuva 1. Laravel 5.2 -tiedostorakenne asennuksen jälkeen

Kun ympäristö on asennettu, on se konfiguroitava käyttöä varten.

2.2 Ympäristön konfigurointi

Ympäristön konfigurointi tapahtuu pääasiallisesti config-kansiossa olevista tiedostoista sekä .env-tiedostosta (kuva 1). Toiminnan kannalta välttämätöntä on konfiguroida web-palveliohjelmiston käyttöoikeudet storage- ja bootstrap/cache-kansioihin.

Palvelin tarvitsee näihin kansioihin kirjoitusoikeuden, joka tapahtuu Linux-ympäristössä antamalla seuraavat oikeudet (5):

```
chown -R www-data:www-data /var/www/html

chmod -R 775 /var/www/html/storage
```

Tämän lisäksi on Laravel vaatii applikaatioavaimen asettamisen .env-tiedostoon (kuva 1). Jos avainta ei aseta, sessiot ja muu enkryptattu data eivät ole suojattuna. Jos Laravel on asennettuna edellisen ohjeen mukaisesti, on avain generoitu automaattisesti. Jos tätä avainta ei jostain syystä ole asetettu, voi sen asettaa Laravelin tarjoaman Artisan-ohjelman avulla (2.):

```
php artisan key:generate
```

Tämä luo 32 merkkiä pitkän applikaatioavaimen, jota käytetään datan enkryptaamiseen.

Laravelin .env-tiedosto tarjoaa myös muita konfigurointi-asetuksia:

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=SomeRandomString
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

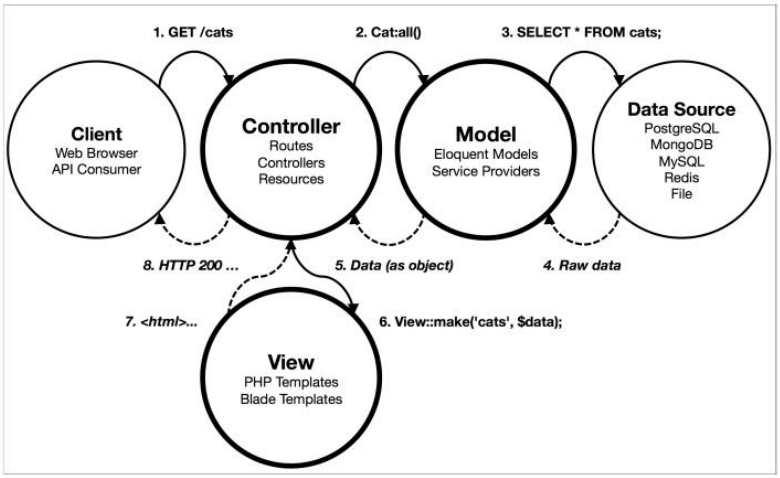
Kuva 2. .env-tiedoston konfigurointimahdollisuudet

Laravel ei vaadi muuta konfigurointia, ja ohjelmistokehitys voidaan aloittaa. Tiedostossa config/app.php löytyy Laravelin toinen tärkeä konfigurointipiste. Tästä tiedostosta pystytään konfiguroimaan seuraavia asioita:

- kehitysympäristö/tuotantoympäristöasetukset
- testausasetukset (engl. Debug settings)
- ohjelmiston URL-asetukset
- aikavyöhyke- ja lokalisaatioasetukset
- enkryptausasetukset
- lokitietojen asetukset
- automaattisten luokkien latauksen määritykset palveluille (engl. autoloaded service providers)
- määrittelemään luokkien aliakset.

3 Laravelin arkkitehtuuri ja ominaisuudet

Laravel noudattaa MVC-suunnittelumallia ohjelmistokehyksessään. Suunnittelumallin toteutumista käytännön projekteissa seurataan yksinkertaisen HTTP-pyynnön polkua ohjelmistokehyksessä.



Kuva 3. HTTP-pyynnön kulkeutuminen Laravelissa (6)

3.1 Malli – Model

Mallin tehtävä ohjelmistossa on kuvata resursseja, joita tallennetaan tietokantaan. Yleisesti mallit vastaavat tietokannassa tietokantatauluja. (6) Malliin liitetään tietokantakenttiä vastaavat luokkamuuttujat sekä pystytään määrittelemään eri luokkien suhteita.

Laravel tarjoaa myös mahdollisuuden nimetä tauluja ja muodostaa automaattisesti yhteyden mallin ja tietokantataulun välillä. Jos tätä nimeämiskäytäntöä ei halua noudattaa, tarjoaa Laravel mahdollisuuden määrittää malli-luokassa vastaavan tietokantataulun, johon tietoa tallennetaan.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Team extends Model
{
    protected $fillable = ['team_name', 'team_color', 'team_category'];

    public function events()
    {
        return $this->hasMany('App\CalendarEvent');
    }
}
```

Kuva 4. Yksinkertainen malli, jota on käytetty tässä esimerkkiprojektissa.

Kuvan 4 esimerkissä määritellään tietokantakenttien nimet fillable -taulukkoon. Tämän lisäksi on määritelty ”monen suhde moneen” (engl. many-to-many relationship) CalendarEvent-luokkaan. Noudattamalla Laravelin nimeämispolitiikkaa osaa laravel yhdistää mallin tietokannssa tauluun *teams*.

Laravelin tarjoaman ORM-tekniikan etuja ovat datan helpon käsittelyn ja koodin luettavuuden parantuminen. Laravel käyttää active record -suunnittelumallia Eloquent ORM-ratkaisussaan. (7) Monimutkaisten tietokantakyselyiden tekeminen Laravelin Query-builderia käyttämällä kuitenkin saattaa olla perinteistä MySQL-käskyä

vaikeampaa, mutta mahdollista. Toiminnon toimintaperiaate selviää parhaiten esimerkistä:

```
$createEvent = new Permission();
$createEvent->name         = 'create-event';
$createEvent->display_name = 'Tapahtumien luominen';
$createEvent->description  = 'Käyttäjä pystyy luomaan uusia tapahtumia.';
$createEvent->save();
```

Kuva 5. Esimerkki Laravel Eloquentin tavasta käsitellä malleja ja niiden tallentamisesta tietokantaan.

Tästä yksinkertaisesta esimerkistä pystytään näkemään, kuinka luettavaa ORM-olioiden tallentaminen tietokantaan on. Laravelin Query Builder tarjoaa myös huomattavan määrän monimutkaisempiin tietokannan tallennus operaatioihin työkaluja, jotka on selkeästi dokumentoitu Laravelin omaan dokumentaatioon.

3.2 Näyttö – View

Laravel tarjoaa MVC-mallin mukaista **V**iew-kerrosta omaa Blade-mallia käyttäen (eng. Blade template). Blade on saanut vaikutteita Razorista, joka toimii näyttökerroksen mallinnuksessa ASP.NET-kirjastossa. (7) Blade tarjoaa tavan rakentaa näyttöjä eri näyttötiedostoista. Blade eroaa myös monesta muusta näyttöjen mallinnus-kirjastosta siinä, että se sallii myös puhtaan PHP:n käyttämisen näyttökerroksessa. Laravel myös tallentaa näyttötiedostoja välimuistiin siihen asti, kunnes niitä on muokattu, mikä vähentää muistinkäyttöä näyttötiedostoissa ja näin lisää ohjelmiston suorituskykyä. (8.)

Blade mahdollistaa näyttötiedostojen hierarkisen rakentamisen. Eri näyttötiedostoja voidaan rakentaa sisäkkäin eri näyttötiedostoista tai näyttötiedostot voivat periä osia toisista näyttötiedostoista, jolloin esimerkiksi muuttujan määrittelyt perityvät suoraan toiseen näyttötiedostoon. (8; 9.)

layouts.app

```
<body id="app-layout">
    <nav class="navbar navbar-default navbar-static-top">
    </nav>

    @yield('content')
</body>
</html>
```

app.list

```
@extends('layouts.app')
@section('content')

<div class="container">
    <div class="row">
```

Kuva 6. Laravelin blade-ominaisuus, jossa injeketoidaan sisältö pääulkoasuun ja määritellään app.list-näytön sisältö.

Laravel sisälsi HTML-helper-kirjaston versiossa 4. Javascript-kirjastojen suosiosta johtuen Laravelista poistettiin versiossa 5 nämä kirjastot. Kirjastoja ylläpitää Laravel Collectiven ”yhteisöorganisaatio”, joksi he kehitystiimiään kutsuvat.

```
<?php echo Form::label('price', 'Hinta', array('class' => 'control-label col-sm-2')) ?>
<div class="col-sm-8">
    <?php echo Form::text('price', '', array('class' => 'form-control')); ?>
</div>
```

Kuva 7. Laravel HTML-helper -työkalu, jolla voidaan määritellä html-komponentteja blade-tiedostoihin.

Kirjaston saa asennettua Composerin avulla Laravel-ohjelmistoon. Kirjasto tarjoaa työkaluja html-elementtien generoimiseen blade-syntaksin avulla. Kirjasto oli suureksi hyödyksi esimerkkiprojektissa, jossa käytettiin staattisia sivuja, joita höystettiin jQuery javascript-kirjastolla.

3.3 Kontrolleri - Controller

Kontrolleri luokkien tarkoituksena on sisältää ohjelmiston logiikka ja ohjata ohjelman suoritusta.

Kuten kuvassa 3 on esitetty, voidaan Laravelin kontrolleriin lukea perinteisten kontrolleri-luokkien lisäksi ohjelmiston reititykseen (engl. routing) tarkoitettu tiedosto sekä resurssit, jotka tarkoittavat reittejä, jotka on määritelty resursseiksi. Tämä ominaisuus määrittelee REST-rajapinnan mukaiset reitit kontrollerin kutsuille. (10.)

Ohjelmiston reititykseen tarkoitettu tiedosto routes.php sisältää kaikki ohjelmiston reitit, joihin palvelin vastaa. Reitteihin voidaan asettaa erilaisia määrityksiä, josta pyynnöt ohjautuvat eteenpäin ohjelmistossa. Reitteihin voidaan myös kytkeä väliohjelmistoja, joka voi olla esimerkiksi käyttöoikeuksien tarkastamista.

```
Route::post('/event/create/{id}', ['middleware' => ['permission:create-event'], 'uses' => 'EventController@create']);
```

Kuva 8. Esimerkki ohjelmiston reitistä, joka sisältää väliohjelmistona käyttöoikeuksien tarkastamista ja pyynnön ohjauksen kontrolleriin.

Laravel ei aseta vaatimuksena ohjata pyyntöä kontrollerille, mutta suurten ja monimutkaisten ohjelmistojen kehityksessä reittien ohjaus kontrollerille on suositeltavaa. Syntaksi on Laravelille ominaisesti hyvin selkeää ja auttaa seuraamaan ohjelmiston logiikan seuraamista.

Kontrolleriin voidaan myös mieltää datan validointiluokat, joita Laravel 5.2 -versiossa voidaan eriyttää kontrollerilta kokonaan, jos kyseessä on lomakkeiden validointia. HTTP-pyynnön (engl. HTTP-request) elinkaari Laravel-ohjelmassa lomakkeiden validoinnissa on seuraava:

1. Asiakasohjelma tekee HTTP-pyynnön
2. Pyyntö ohjautuu reititykseen
3. Jos validointiluokka on luotu ohjautuu pyyntö validaattorille

4. Validaattori validoi datan, jota pyynnössä toimitetaan

5. Pyyntö ohjautuu kontrollerille

6. Kontrolleri tekee pyynnön käsittelyn ja mahdollisesti palauttaa uuden näytön asiakasohjelmalle.

Tässä työssä käytettiin paljon erilaisia lomakkeita, joiden datan validointiin lähes poikkeuksetta käytettiin Laravelin validaattoriluokkia. Laravel tarjoaa datan validointiin kattavat työkalut perustoiminnallisuudessaan. Näiden lisäksi pystytään kirjoittamaan omia validointisääntöjä tarvittaessa Laravelin validator facadea käyttämällä. (11.) Laravelin validaattorin säännöt määritellään rules -taulukkoon, jossa kerrotaan avain-arvo-pareilla odotettu arvo.

```php
public function authorize()
{
    return true;
}
/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'calendar_name' => 'required|string',
        'calendar_code' => 'required|string',
        'calendar_category' => 'required|string',
    ];
}
```

Kuva 9. Validointisääntöjen määrittely CalendarRequest-luokkaan.

3.3 Migraatiot ja tietokannan hallinta

3.3.1 Migraatiot

Laravel tarjoaa tietokannan hallintaan oman migraatiotyökalun. Migraatioiden avulla voidaan hallita tietokannan tauluja ja kenttiä. Migraatiot toimivat eräänlaisena versionhallintana tietokannalle, josta voidaan helposti lisäillä uusia kenttiä tai palauttaa tietokanta tiettyyn pisteeseen. (12.)

Migraatioiden avuilla pysytään myös vaihtamaan tietokannanhallintajärjestelmää (TKHJ) tarvittaessa uuteen helposti. Laravelissa voidaan konfiguroida uuden tietokannan tiedot ja suorittaa migraatiot, jotka nostavat tietokantarakenteen uuden TKHJ:n sisälle.

Migraatiot voidaan luoda käyttämällä Laravelin artisan-työkalua. Uusi migraatio luodaan suorittamalla komentoriviltä käsky

```
php artisan make:migration create_users_table
```

Käskystä viimeinen osa toimii migraation nimenä. Laravel lisää myös aikaleiman jokaisen migraation nimen etuliitteeksi, jotta Laravel osaa järjestää migraatiot oikeaan aikajärjestykseen. (12.)

Migraatiot tallentuvat automaattisesti kansioon database/migrations, mutta polkua voidaan myös vaihtaa antamalla käskyyn path-parametri. Migraatiolle voidaan myös antaa muita parametreja, kuten esimerkiksi –table, joka generoi automaattisesti migraatioon halutun tietokantataulun, johon tietokannan muutos kohdistuu. (12.)

Migraatiotiedosto sisältää kaksi funktiota, jotka ovat *up* ja *down*. Nämä funktiot määrittelevät migraation käyttäytymistä, kun migraatio ajetaan ja kun migraatio palautetaan (engl. rollback).

```php
/**
 * Run the migrations.
 *
 * @return void
 */
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name');
        $table->string('email')->unique();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::drop('users');
}
```

Kuva 10. Tietokantaan users-taulun luominen Laravelin migraation avulla.

Laravel tarjoaa tietokannan taulujen kenttien määritykseen hyvin selkeän ja luettavan tavan luoda uusia kenttiä ja niiden suhteita toisiin tietokantatauluihin. Eri kenttien merkitykset tietokantaan on dokumentoitu kattavasti Laravelin dokumentaatioon.

Tietokannan tauluja voidaan migraatioiden avulla luoda, muuttaa, poistaa tai muokata. On hyvä huomioida kaikki operaatiot, joita suoritetaan, on lisättävä omiin migraatiotiedostoihin, jotta Laravel pysyy migraatioiden tilasta tietoisena. Muutoksia ei siis tehdä olemassa oleviin migraatioihin vaan niitä varten generoidaan uusi migraatio.

3.3.2 Migraatiot muissa ohjelmistokehyksissä

Migraatiot on huomattu yleisesti hyväksi tavaksi hallinoida tietokantaa ja niiden kenttiä, ja ominaisuus on myös käytössä muissa suosituissa ohjelmistokehyksissä, jotka eivät rajoitu PHP-skriptikieleen.

Toimintaperiaatteiltaan migraatiot ja niiden hallinta esiintyy hyvin samankaltaisina monissa tunnetuissa ohjelmistokehyksissä kuten Python-skriptikieleen perustuvassa Djangossa tai Ruby-kieleen perustuvassa Ruby on Railsissä.

Periaatteiltaan toteutukset ovat hyvin samanlaisia, mutta eroavaisuuksia löytyy. Djangossa migraatiot on kytketty suoraan malleihin, jotka on eroteltu Laravelissa migraatioista. (13.)

```python
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

Kuva 11. Mallin määrittäminen Django-ohjelmistokehyksessä.

Ruby on Railsissä migraatioiden luonti toimii hyvin samalla periaatteella kuin Laravelissa, jossa jopa funktioiden nimet ovat samat.

```ruby
class AddSsl < ActiveRecord::Migration[5.0]
  def up
    add_column :accounts, :ssl_enabled, :boolean, default: true
  end

  def down
    remove_column :accounts, :ssl_enabled
  end
end
```

Kuva 12. Migraatio Ruby on Rails -ohjelmistokehyksessä. (14)

3.3.3 Tietokannan kylväminen (engl. Database seeding)

Tietokannan kylväminen on ominaisuus, jolla voidaan syöttää tietokantaan testidataa. (15) Tässä insinöörityössä käytetyssä esimerkkiprojektissa käytettiin tietokannan kylvämistä käyttöoikeuksien syöttämiseen tietokantaan.

Tietokannan kylväjä voidaan generoida artisan-työkalulla:

```
php artisan make:seeder UsersTableSeeder
```

Komento generoi tiedoston database/seeds –kansioon. Tiedosto sisältää luokan, jossa on yksi funktio *run*. Tämän funktion sisään voidaan tehdä ne operaatiot, joita tietokantaan halutaan syöttää. Tietokannan kylväjä ajetaan komennolla:

```
php artisan seed
```

Parametreiksi voidaan antaa –class-parametri, jolla voidaan osoittaa oikea luokka, joka halutaan käyttää tietokannan kylvämiseen.

```php
$superuser = new Role();
$superuser->name         = 'SuperUser';
$superuser->display_name = 'Ylläpitäjä'; // optional
$superuser->description  = 'Ylläpitäjän rooli antaa kaikki oikeudet järjestelmän kaikkiin polkuihin.'; // optional
$superuser->save();
```

Kuva 13. Esimerkki tietokannan kylväjän ajamasta datasta.

Tietokannan kylväjästä voidaan myös kutsua muita tietokannan kylväjiä käyttämällä call-metodia sekä voidaan generoida suuria määriä dataa käyttämällä mallitehtaita (engl. model factory)

3.4 Artisan

Laravelin Artisan-työkalu tarjoaa komentorivipohjaisen työkalun, jonka avulla voidaan tehdä monia rutiinitoimenpiteitä suoraan komentorivillä. Esimerkiksi Laravelin kontrolleriluokkien rungot voidaan generoida suoraan artisania käyttämällä. Tämä säästää aikaa tiedostojen kopioimiselta tai puhtaasti uusien tiedostojen kirjoittamiselta.

Artisan tarjoaa myös REPL-työkalun (Read-eval-print loop), joka kantaa nimeä Tinker. Tinkerin avulla voidaan olla suorassa vuorovaikutuksessa ohjelmiston kanssa, joka mahdollistaa komentoriviltä esimerkiksi uusien käyttäjien luonnin ohjelmistoon.

Artisan-työkalu sisältää sisäänrakennettuna paljon erilaisia elämää helpottavia komentoja, joita on mahdollista myös tehdä itse. Tätä varten Laravel on helpottanut

komentorivillä suoritettavien komentojen tekemistä kirjastojen avulla. Tämä on kattavasti dokumentoitu Laravelin omassa käyttöoppaassa. (16.)

4 Tilavarausjärjestelmä urheiluseuroille

4.1 Tavoite

Tavoitteena on rakentaa minimiominaisuuksilla toimiva tilavarausjärjestelmä, joka perustuu kokemuksiin urheiluseurojen monipuolisista tarpeista joukkueiden varausten hallintaan.

Urheiluseurat hallinoivat nykypäivänä varauksiaan pääasiallisesti Excelillä. Varausten tuomaa informaatiota on kuitenkin hankalaa jalostaa. Tämän työn tavoite on luoda tätä tarkoitusta varten esimerkkiohjelmisto, jolla pystytään suorittamaan perustoimenpiteitä varausten luonnista muokkaukseen ja niiden seuraamiseen.

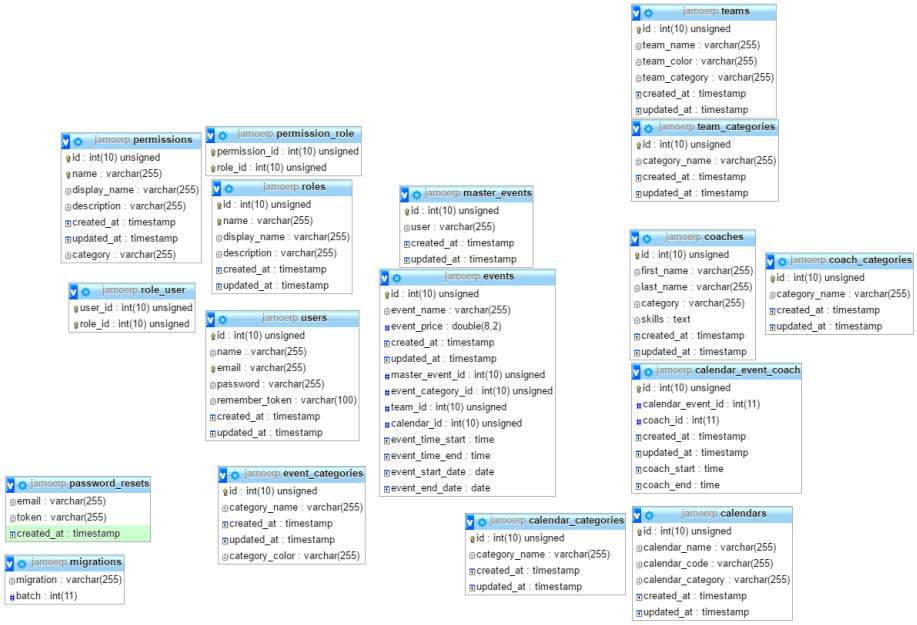
Työn lopputuloksena järjestelmästä löytyvät seuraavat ominaisuudet:

- kalentereiden, joukkueiden, valmentajien luonti
- erilaiset kategorisoinnit (mm. joukkueen ja varauksen kategorisointi)
- raportointi
- käyttäjien hallinta
- varausten luonti kalenteriin ja niiden muokkaaminen.

4.2 Järjestelmän arkkitehtuuri

4.2.1 Tietokanta

Tietokantaohjelmistoksi valittiin MySQL, jota Laravel tukee suoraan asennuksesta. Tietokanta koostuu yhteensä 17 tietokantataulusta, joista osa muodostaa selkeitä tietoja sisältäviä tauluja ja osa toimii yhdistävinä välitauluina, joihin merkitään eri taulujen suhteita.



Kuva 14. Järjestelmässä käytettävät tietokantataulut.

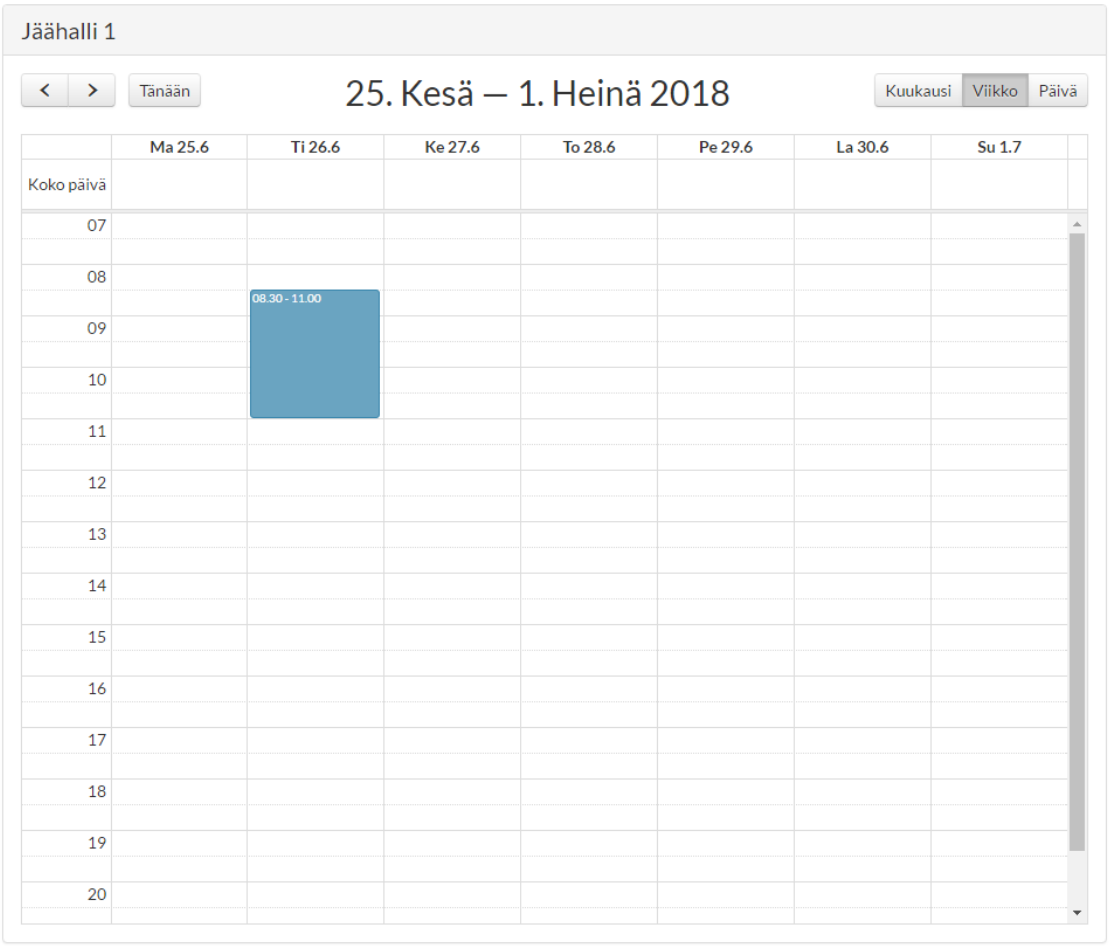
Toteutuksen jälkeen todettiin, että olisi ollut järkevämpää tiivistää kategorisointi yhteen tauluun, jolloin olisi myös mahdollista käyttää yhtä näkymää kategorisoinnin tallentamiseen sen eri merkityksissä. Tämä on tulevaisuudessa järkevä toteuttaa. Se poistaa kompleksisuutta järjestelmästä ja helpottaa järjestelmän ylläpitoa niin teknisesti kuin käytettävyydeltään.

4.3 Toiminnot

4.3.1 Varausten tekeminen järjestelmään

Varaukset tehdään kiinnittämällä varaukset eri kalentereihin. Kalenterit vastaavat käytännössä esimerkiksi kenttiä tai erilaisia tiloja, joihin varaus asetetaan. Varaukset kiinitetään aina yhteen varauskohteeseen kerrallaan. Varauksiin ei ole toistaiseksi mahdollista poimia monia tiloja erikseen. Tämäntyyppinen varaaminen on kuitenkin hyvin harvinaista käytännössä.

Varauksen luominen järjestelmään aloitetaan maalamalla Fullcalendar-kirjaston tarjoamaan kalenterinäkymään varaus:



Kuva 15. Varauksen maalaminen kalenteriin esimerkkiprojektissa.

Fullcalendar-integrointiin on käytetty Maddhatter/laravel-fullcalendar-kirjastoa, joka mahdollistaa kalenterin generoinnin PHP:n avulla (https://github.com/maddhatter/laravel-fullcalendar). Fullcalendar on javascript-kirjasto, joka mahdollistaa modernin kalenterinäkymän luomisen. Toiminta perustuu dataan, joka annetaan Fullcalendarille ns. tapahtumina (eng. events). Fullcalendar hoitaa tapahtumien renderöinnin webselaimessa.

Laraveliin tehdyn Maddhatter/laravel-fullcalendar-kirjaston tarkoituksena on helpottaa datan siirtämistä fullcalendar-kirjastolle kontrolleriluokista. Tämän lisäksi pystytään myös antamaan asetuksia sekä soittopyyntö metodien (engl. callback methods) määrityksiä, joita Fullcalendar tukee.

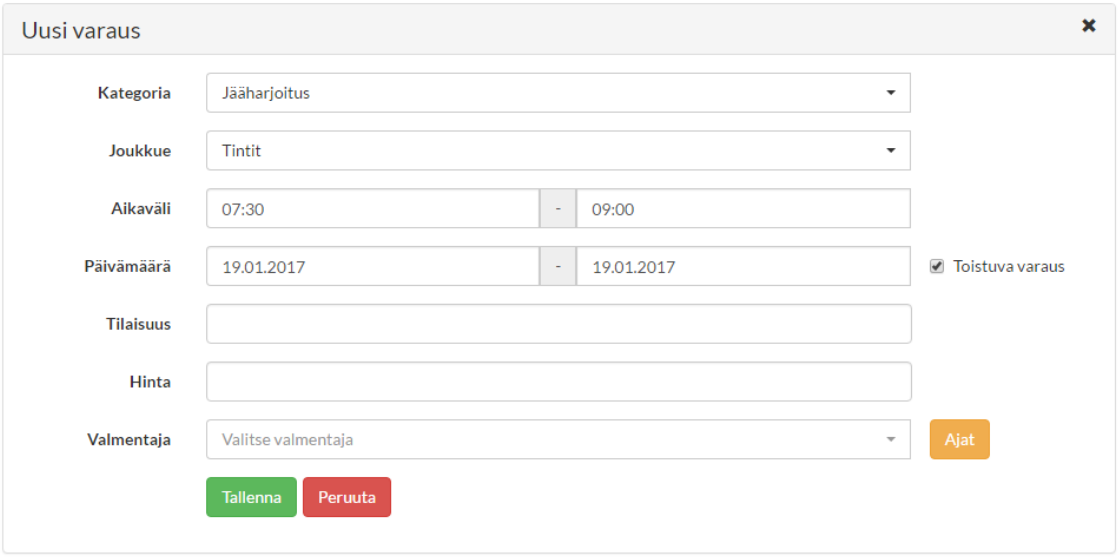
```php
$events = [];

$events[] = \Calendar::event(
    'Event One', //event title
    false, //full day event?
    '2015-02-11T0800', //start time (you can also use Carbon instead of DateTime)
    '2015-02-12T0800', //end time (you can also use Carbon instead of DateTime)
    0 //optionally, you can specify an event ID
);

$events[] = \Calendar::event(
    "Valentine's Day", //event title
    true, //full day event?
    new \DateTime('2015-02-14'), //start time (you can also use Carbon instead of DateTime)
    new \DateTime('2015-02-14'), //end time (you can also use Carbon instead of DateTime)
    'stringEventId' //optionally, you can specify an event ID
);

$eloquentEvent = EventModel::first(); //EventModel implements MaddHatter\LaravelFullcalendar\Event

$calendar = \Calendar::addEvents($events) //add an array with addEvents
    ->addEvent($eloquentEvent, [ //set custom color fo this event
        'color' => '#800',
    ])->setOptions([ //set fullcalendar options
        'firstDay' => 1
    ])->setCallbacks([ //set fullcalendar callback options (will not be JSON encoded)
        'viewRender' => 'function() {alert("Callbacks!");}'
    ]);

return view('hello', compact('calendar'));
```

Kuva 16. Esimerkkikoodi kontrollerin toiminnasta ja tapahtumien lisäämisestä fullcalendariin käyttäen Maddhatter/laravel-fullcalendar-kirjastoa. (17)

Varauksia voidaan luoda järjestelmään kahdella eri tavalla: Yksittäisiä tai toistuvia varauksia. Toistuvat varaukset toistuvat viikoittain samana ajankohtana ja yksittäiset varaukset ovat kertaluonteisia tapahtumia, jotka merkataan kalenteriin.

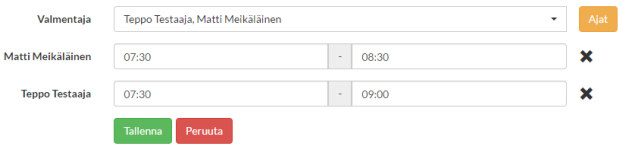
Toistuvaa varausta varten tietokantaan tehtiin master_event-taulu, jonka tarkoituksena on yhdistää yksittäiset varaukset yhdeksi varaustapahtumaksi. Tällä menetelmällä pystytään muokkaamaan viikoittain toistuvia varauksia samasta näkymästä sen sijaan, että niitä muokattaisiin yksi kerrallaan.

Varaus sisältää monipuolisesti dataa, jota pystytään hyödyntämään raportoinnissa. Näitä ovat mm. tapahtuman kategoria, joukkue ja aikaväli.



Kuva 17. Varauksen luonnin lomake.

Varaukseen pystytään myös kiinnittämään dynaamisesti valmentajia. Valmentajat tuodaan järjestelmään kirjattujen valmentajien tietokantataulusta. Valmentajia voi valita haluamansa määrän ja heidän läsnäolonsa varaukseen voidaan muokata vapaasti.



Kuva 18. Valmentajien kiinnitys varaukseen

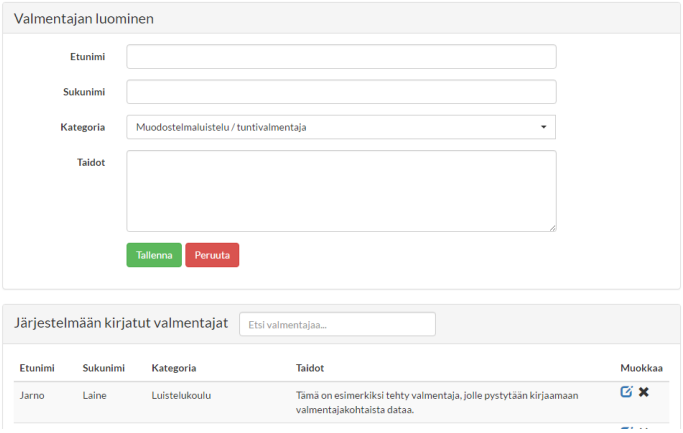
Valmentajille ei ole asetettu aikarajoitteita varaukselle, sillä he saattavat olla aikasemmin paikalla tai olla varausta pitempään valmentamassa. Myös valmentaja saattaa olla osan toisessa varauskohteessa, jossa varaus voi kestää pitempään. Tunnit kuitenkin voidaan jyvittää, kuten varausten tekijä parhaaksi näkee.

Tulevaisuuden kehityskohteina varauksien tekemisessä järjestelmään voisi olla varausten peruminen. Tämä peruminen voitaisiin hoitaa esimerkiksi erilaisilla vahvistustiloilla, kuten vahvistettu tai peruttu. Tämän jälkeen ominaisuus pitäisi pystyä nostamaan mukaan myös raportointiin, jossa voitaisiin rajata tarkastelu vahvistustilan perusteella, jolloin saataisiin erilaisten peruutusten vaikutus varauksiin ja näitä varauksia ei tarvitsisi kokonaan poistaa tietokannasta.

4.3.2 Kategorioiden luonti

Järjestelmä antaa monipuoliset välineet kategorisoida erilaisia resursseja, kuten kalentereita, joukkueita ja tapahtumia. Kategorisoinnin ideana on antaa käyttäjälle mahdollisuus suodattaa raportoinnissa varauksista haluamaansa dataa.

Kategorioiden avulla pystytään urheiluseuroissa tarkastelemaan esimerkiksi valmentajien tuntien jakautumista eri kategorioiden avulla. Käytännössä tämä voisi tarkoittaa esimerkiksi taitoluistelussa valmentajan kuulumista eri jaostoon (muodostelmaluistelu, lusitelukoulu, yksinluistelu jne.). Näin voidaan jaotella valmentajat eri jaostoihin kategorisoinnin avulla.



Kuva 19. Valmentajan luonti järjestelmään, jossa voidaan valita valmentajan kategoria pudotusvalikosta.

Kategorisointi toteutettiin järjestelmään luomalla eri tietokantatauluja vastamaan kategorisointia ja tuottamalla mallit ja määritettiin eri kategorioiden suhteet varauksiin. Kategorisoinnissa käytettiin periaatteena, että joukkue, tapahtuma tai varaus voidaan kategorisoida yhteen kategoriaan kerrallaan. Tämän toteutusmallin hyvä puoli oli sen helppo toteutus, mutta käytännössä saattaa tulla vastaan tilanteita, jossa moneen eri kategoriaan kiinnittäminen voisi olla hyödyllistä. Toisaalta tämä aiheuttaa kompleksisuutta raportoinnin kanssa ja lisää monimutkaisuutta järjestelmän käytössä.

Tästä hyvänä esimerkkinä voidaan pitää valmentajan kategorisointia. Valmentaja saattaa käytännössä valmentaa tai sijaistaa montaa eri joukkuetta, jotka kuuluvat tiettyjen kategorisointien alle. Tämä vaatisi järjestelmän tasolta kategorisointia moneen

eri kategoriaan, jonka lisäksi pitäisi kytkeä valmentajien kategoriat vastaamaan tapahtumissa esiintyviä kategorioita, joista voitaisiin päätellä, mihin kategoriaan valmentajan tunnit kohdistuvat.

Tämä ei kuitenkaan käytännön tilanteissa ole noussut mitenkään merkittäväksi asiaksi, mutta on huomionarvoinen vivahde, jolla järjestelmää saataisiin monipuolistettua.

4.3.3 Raportointi

Raportoinnin avulla voidaan suodattaa varausten dataa ja saada statistiikkaa siihen liittyvistä komponenteista, kuten valmentajista tai tapahtumista. Raportointiin on tehty kaksi erilaista jaottelua, jolla voidaan tarkastella varauksia joko valmentajan näkökulmasta tai tapahtumien näkökulmasta.

Raportointi tukee seuraavia yleisiä raportoinnin tarpeita urheiluseuroissa:

- tapahtumien kustannukset tietyllä aikavälillä sekä kellonaika rajauksella
- joukkueiden tuntien jakautuminen
- joukkueiden kustannukset
- harjoitusten lukumäärä
- kategorisoinnin tuomat suodatukset, kuten esimerkiksi tuntien jakautuminen eri kategorisointien kesken
- valmentajien tuntien seuranta tapahtumien perusteella.

Raportointiin on käytännössä nostettu kaikki mahdolliset tiedot, joita varaukseen on kirjattu, jolloin tätä dataa voidaan hyödyntää tarvittaessa.

Teknisessä mielessä raportointi on datan hakua asetetuilla parametreillä tietokannasta. Apuna on käytetty datan renderöimiseen DataTables-nimistä jQueryyn pohjautuvaa javascript-kirjastoa. DataTables (https://www.datatables.net) muokkaa näyttötiedostoon piiretyn taulukon javascriptin avulla muotoon, jossa voidaan järjestellä dataa haluttuun järjestykseen ja tarjoaa valmiit työkalut esimerkiksi numeeristen arvojen manipulointiin.

Raportoinnista tehtiin hyvin pelkistetty versio, joka tarjoaa hyvän pohjan jatkojalostukselle. Käytännön kokemuksiin vastaavien projektien toimituksessa on havaittu, että raporttien valmislistaukset ovat yksi tärkeimmistä raportointia helpottavista työkaluista, jolloin raportointityökalun opettelmiseen ei tarvitse uhrata aikaa vaan raportointitarpeet voitaisiin tallentaa valmiiksi järjestelmään. Tätä ominaisuutta ei vielä toteutettu tässä projektissa.

4.3.4 Autentikointi ja käyttöoikeuksien hallinta

Autentikointi järjestelmiin on yksi useimmin tehtävistä asioista, jonka itse tekeminen on tietoturvan näkökannalta erittäin haastavaa. Laravel helpottaa autentikoinnin käyttöönottoa, joka on tulee Laravel-asennuksen mukana. Laravelin asennukseen on konfiguroitu autentikointiin vaadittavat osat valmiiksi. (18.)

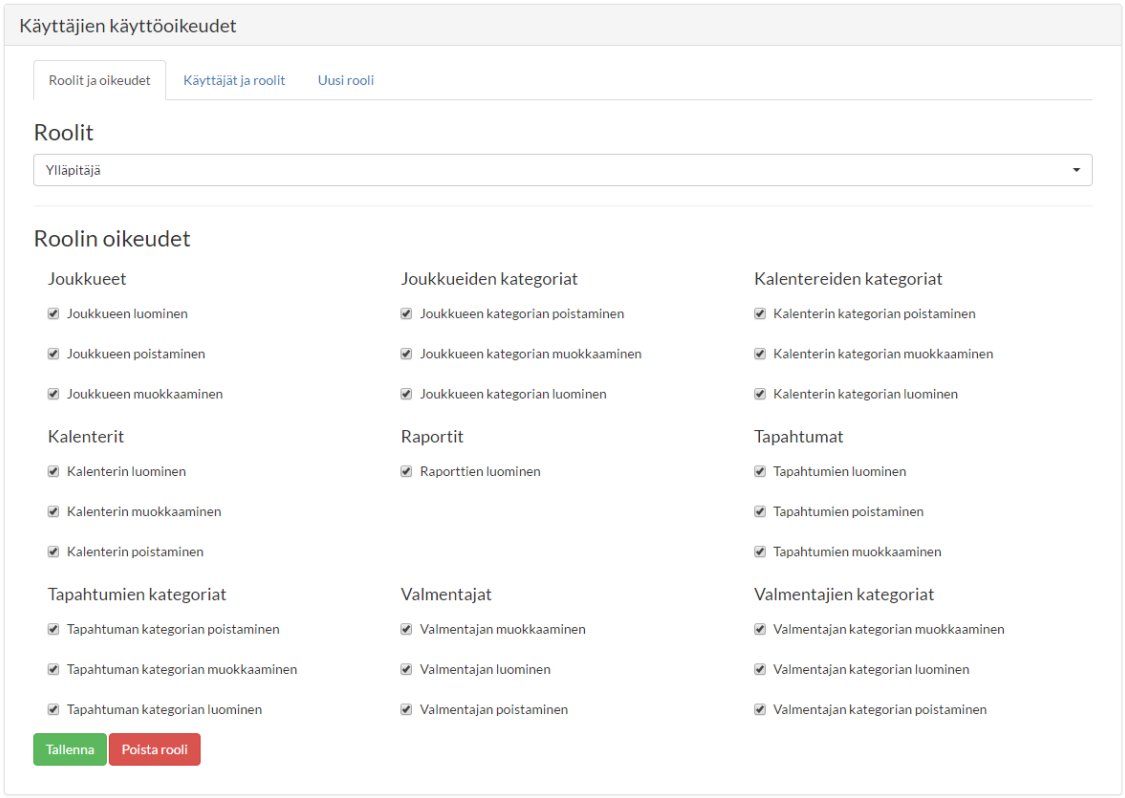
Valmiita asetuksia pystytään konfiguroimaan tiedostosta config/auth.php, josta voidaan säätää esimerkiksi, pidetäänkö kirjautumista sessioissa vai autentikaatio tokeneissa sekä konfiguroida salasanan resetoimiseen tarkoitetun url-osoitteen voimassaoloaikaa.

Autentikaation perusversion käyttöön voidaan käyttää seuraavaa komentoa, joka luo rekisteröintilomakkeet, sisäänkirjautumisen ja tarvittavat reitit kaikkiin autentikaation osiin:

```
php artisan make:auth
```

Autentikointia kustomoitiin tähän projektiin ainoastaan kääntämällä englanninkielisiä termejä suomeksi.

Käyttöoikeuksien hallintaa Laravel ei toimita valmiiksi asennettuna. Tätä varten asennettiin ulkoinen kirjasto Entrust, joka on Laraveliin suunniteltu avoimen lähdekoodin toteutus. Asennus ja konfigurointi on opastettu vaiheittain heidän github-sivullaan https://github.com/Zizaco/entrust.



Kuva 20. Käyttöliittymästä esimerkki rooliin asetettavista käyttöoikeuksista

Entrustin toimintaperiaatteena on roolipohjainen käyttöoikeuksien hallinta. Entrust toimii Laraveliin asennetun autentikaation kanssa ja vaatii users-taulun tietokannasta sekä vastaavan Eloquent-mallin users-taulusta, johon Entrustin luomat roolit kytkeytyvät php-traitin avulla.

Entrustissa on yksittäisiä käyttöoikeuksia, jotka kiinitetään käyttöoikeusrooliin. Rooli kiinnitetään käyttäjään, jonka avulla pystytään muodostamaan käyttäjäryhmiä käyttötarpeiden mukaisesti ja asettamaan eri käyttäjäryhmille omia rooleja.

Projektia varten toteutettiin käyttäjien hallinta, johon lisättiin projektissa käytettyihin reitteihin perustuvat käyttöoikeudet. Entrust tarjoaa middlewaren, jonka avulla jokaiseen reittiin pystytään konfiguroimaan käyttöoikeusmäärittelyt perustuen yksittäiseen käyttöoikeuteen.

```
Route::get('/event/{id}', ['as' => 'event.event', 'uses' => 'EventController@index']);
Route::post('/event/create/{id}', ['middleware' => ['permission:create-event'], 'uses' => 'EventController@create']);
Route::get('/event/edit/{id}', ['as' => 'event.edit', 'middleware' => ['permission:edit-event'], 'uses' => 'EventController@edit']);
Route::get('/event/delete/{id}', ['middleware' => ['permission:delete-event'], 'uses' => 'EventController@delete']);
Route::get('/event/delete/master/{id}', ['middleware' => ['permission:delete-event'], 'uses' => 'EventController@deleteMaster']);
Route::post('/event/update/{id}', ['middleware' => ['permission:edit-event'], 'uses' => 'EventController@update']);
```

Kuva 21. Esimerkki routes.php-tiedostoon asetetuista käyttöoikeuksien määrittelystä koskien ohjelman polkuja.

5 Yhteenveto

Tarkoituksena oli oppia Laravel-ohjelmistokehyksen toimintaa ja sen hyödyntämistä käytännön projekteissa. Esimerkkiprojektissa käytettiin Laravelin tarjoamia perustoiminnallisuuksia, jotka helpottivat ja nopeuttivat huomattavasti ohjelmiston kehitystä verrattuna täysin puhtaalta pöydältä aloittamiseen.

Tämän lisäksi ohjelmistossa käytettiin paljon ulkoisia kirjastoja, jotka olivat Laravelin kanssa yhteensopivia. Kirjastojen käyttöönotto nopeutti ohjelmiston kehitystä ja siirsi kehityksen painopisteen ohjelmiston perustoiminnallisuuksiin.

Aktiivisen kehitystiimin sekä yhteisön kehittelemien apukirjastojen avulla aikaa vievien perustoiminnallisuuksien kehittäminen oli todella nopeaa.

Esimerkkiprojekti tarjosi myös hyvän kuvan ohjelmiston arkkitehtuurillisten ratkaisujen vaikutuksesta ohjelmistokehitykseen. Esimerkkiprojektia kehitetään edelleen vastamaan täysin urheiluseurojen tarpeita sekä tarkoituksena on päivittää ohjelmistokehys uusimpaan Laravel 5.4 -versioon.

Lähteet

1 Wikipedia. 2017. Verkkojulkaisu. https://en.wikipedia.org/wiki/Laravel. Luettu 14.2.2017.

2 Installation. 2017. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/installation. Luettu 14.2.2017.

3 PHP Archive. 2015 PHP Documentation. http://php.net/archive/2015.php. Luettu 21.4.2017.

4 Database: Getting Started. 2017. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/database. Luettu: 16.2.2017.

5 How-to Install PHP 7.x, NGINX 1.9.x & Laravel 5.x. William. Asked.io. Verkkojulkaisu. https://asked.io/how-to-install-php-7-x--nginx-1-9-x---laravel-5-x. Luettu 24.2.2017.

6 Bean, Martin. Huhtikuu 2017. Packt Publishing Ltd. s.35 Laravel 5 Essentials.

7 Bean, Martin. Huhtikuu 2017. Packt Publishing Ltd. s.32 Laravel 5 Essentials.

8 Blade Templates. 2017. Laravel Documentation https://laravel.com/docs/5.2/blade. Luettu 25.01.2017.

9 Bean, Martin. Huhtikuu 2017. Packt Publishing Ltd. s.69 Laravel 5 Essentials.

10 Controllers. 2017. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/controllers. Luettu 12.2.2017.

11 Custom Validation Rules. 2017. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/validation#custom-validation-rules. Luettu 05.03.2017.

12 Database: Migrations. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/migrations. Luettu 18.03.2017.

13 Migrations. Django Documentation. Verkkojulkaisu. https://docs.djangoproject.com/en/1.10/topics/migrations. Luettu 18.03.2017.

14 Active Record Migrations. Ruby on Rails 5.0.2 API. Verkkojulkaisu. http://api.rubyonrails.org/classes/ActiveRecord/Migration.html. Luettu 18.03.2017.

15 Database: Seeding. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/seeding. Luettu 18.03.2017.

16 Artisan Console. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/artisan. Luettu 19.03.2017

17 Maddhatter/laravel-fullcalendar. Github. Verkkojulkaisu. https://github.com/maddhatter/laravel-fullcalendar. Luettu 18.01.2017

18 Authentication. Laravel Documentation. Verkkojulkaisu. https://laravel.com/docs/5.2/authentication. Luettu 19.02.2017